Microcode

From Wikipedia, the free encyclopedia

A **microprogram** implements a CPU instruction set. Just as a single high level language statement is compiled to a series of machine instructions (load, store, shift, etc), in a CPU using microcode, each machine instruction is in turn implemented by a series of microinstructions, sometimes called a microprogram. Microprograms are often referred to as **microcode.**

The elements composing a microprogram exist on a lower conceptual level than the more familiar assembler instructions. Each element is differentiated by the "micro" prefix to avoid confusion: microprogram, microcode, microinstruction, microassembler, etc.

Microprograms are carefully designed and optimized for the fastest possible execution, since a slow microprogram would yield a slow machine instruction which would in turn cause all programs using that instruction to be slow. The microprogrammer must have extensive low-level hardware knowledge of the computer circuitry, as the microcode controls this. The microcode is written by the CPU engineer during the design phase.

On most computers using microcode, the microcode doesn't reside in the main system memory, but exists in a special high speed memory, called the control store. This memory might be read-only memory, or it might be read-write memory, in which case the microcode would be loaded into the control store from some other storage medium as part of the initialization of the CPU. If the microcode is read-write memory, it can be altered to correct bugs in the instruction set, or to implement new machine instructions. Microcode can also allow one computer microarchitecture to emulate another, usually more-complex architecture.

Microprograms consist of series of microinstructions. These microinstructions control the CPU at a very fundamental level. For example, a single typical microinstruction might specify the following operations:

- Connect Register 1 to the "A" side of the ALU
- Connect Register 7 to the "B" side of the ALU
- Set the ALU to perform two's-complement addition
- Set the ALU's carry input to zero
- Store the result value in Register 8
- Update the "condition codes" with the ALU status flags ("Negative", "Zero", "Overflow", and "Carry")
- Microjump to MicroPC nnn for the next microinstruction

To simultaneously control all of these features, the microinstruction is often very wide, for example, 56 bits or more.

Contents